

### **REMARKS**

Claims 1, 3-5 and 11-16 are pending and under consideration in this application. Claims 1, 11, and 14 are amended herein. Claims 15 and 16 are added herein. Support for the amendments to the claims may be found in the claims as filed originally. Reconsideration is requested based on the foregoing amendment and the following remarks.

#### **Response to Arguments:**

The Applicants appreciate the consideration given to their arguments. The Applicants, however, were disappointed to find that their arguments were not found to be persuasive. The final Office Action asserts in section A, at the bottom of page 8, continuing at the top of page 9, that:

The Examiner respectfully disagrees because lms specifies the input properties (i.e. categories or book titles) are used in retrieving available book inventory information (lms, col. 10, line 51-53). In other words, lms' input properties are essentially values used in the retrieval of information. Because the input properties of lms are used to retrieve information (i.e. col. 10 line 55, lms states getting results with the input properties), they specify a retrieval condition that must be met to return results.

This is submitted to be incorrect. Neither a category name nor a book title, both of which are given as examples of input properties at column 10, lines 51 and 52 of lms, are equivalent to a retrieval *condition*, whether or not they are used in retrieving available book inventory information. The input properties of lms, rather, are used to retrieve information, as noted in the final Office Action.

lms himself, in fact, uses the word "condition" differently than the interpretation given to it in the final Office Action. lms, for example, enables writes to be scheduled for processing when particular events occur or when particular system-specific *conditions* are met. In particular, as described at column 19, lines 39-44:

For applications which are adaptable to this technique, this aspect of the present invention enables better resource utilization to occur by avoiding unnecessary reads to a back-end data source and by enabling writes to be scheduled for processing when particular events occur or when particular system-specific conditions are met.

Thus, lms defines a condition as something to be met, rather than values used in the retrieval of information. lms, consequently, which is cited in the final Office Action as evidence of

skill in the art, does not support the interpretation of a "retrieval condition" to which the final Office Action adheres.

The final Office Action goes on to assert in section A, in the first full paragraph at page 9, that:

Furthermore the Examiner submits that from review of Applicant's own disclosure (page 9, line 8-9), which specifies "using the retrieval conditions (the SQL retrieval expression)", this interpretation is reasonable. In other words, Applicant's retrieval condition can be a SQL expression for accessing a database. Likewise, since lms' input properties specify what to retrieve from a database, they can readily be seen as Applicant's retrieval condition (i.e. a retrieval statement).

The subject specification describes the SQL retrieval expression as based on the retrieval condition at page 7, line 18. Moreover, as described at page 8 of the subject specification, in lines 7, 8, and 9:

When the terminal 1 transmits a retrieval condition, the application server 30 produces an SQL retrieval expression based on the retrieval condition.

Neither a category name nor a book title, on the other hand, both of which are given as examples of input properties at column 10, lines 51 and 52 of lms, are equivalent to an SQL retrieval expression, let alone a retrieval *condition*.

The final Office Action asserts finally in section A, in the second full paragraph at page 9, that:

Subsequently, Applicant argues that the word "condition" is used in lms' differently than the interpretation given to it in the Office Action. The Examiner submits that there was no interpretation given to this portion (i.e. col. 19 lines 39-44) in the previous Office Action and this aspect was not relied upon to reject the retrieval condition. Therefore this argument is not found to be pertinent to this discussion and further moot.

The passage provided was to indicate that the interpretation of the word "condition" used by lms, which was cited in the final Office Action as evidence of one of skill in the art, conflicts with the interpretation given to the word "condition" in the final Office Action. Thus, even though the passage cited was not cited in the final Office Action, it is still relevant, since it shows that lms himself, which is cited as evidence of skill in the art at the time the invention was made, used the word "condition" differently than the interpretation adopted in the final Office Action.

The final Office Action asserts in section B, at the top of page 10, that:

Because these properties are stored together in the instance of a bean, they are stored in correlated form.

This is submitted to be without basis. Whether properties are stored *together* or not has no bearing on whether they are stored in correlated or, for that matter, uncorrelated form.

The final Office Action asserts further in section B, in the first full paragraph at page 10, that:

In response to the reply, the Examiner further clarifies the position taken in this aspect of storing, in correlated form, the retrieval condition and retrieval result. As previously mentioned, the input properties (retrieval condition) and retrieval results (results of the lookup operation using the input properties; col. 10 line 54-56) at least describe the broadly claimed "in correlated form". In other words, Applicant claims no further details as to specify what "in a correlated" form is to precisely comprise. Therefore with the interpretation of storing these two aspects together in an object (i.e. a bean), this object gives the two elements (input/output properties) a relation (or "correlation"), and thus this limitation is taught.

This is submitted to be incorrect. Two aspects could be stored randomly together in an object more easily than they could be stored in correlated form. Nor is there any reason for lms to bother storing the input properties and retrieval results together in correlated form since, as discussed more fully below, lms is using the input properties and values thereof as an index to *locate* objects stored in the cached objects component, not as a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11.

Finally, the final Office Action asserts in section B, in the second full paragraph at page 10, that:

Moreover, lms teaches "in correlated form" in another example (e.g. col. 14 line 51-57). That is, lms teaches a refreshing of a cached object wherein retrieval logic in the execution script of the cached object (col. 14 line 52-53) which are used to refresh (i.e. re-populating) the objects output properties (col. 14 line 56-57). Put another way, since the retrieval logic in a cached object is used to refresh that same object's output properties (results), the object stores the retrieval condition and result in correlated form.

This is submitted to be incorrect. In lms, rather, there is retrieval *logic* in the execution script of the cached object, not a "retrieval condition" or a "retrieval result" "stored in a correlated form" as recited in, for example, claims 1 and 11. In particular, as described at column 14, lines 51-62:

For example, cached object HAO2 shown at 312 is refreshed by executing the retrieval logic in the execution script of the cached object. This retrieval logic causes a request to be sent 303 to the back-end data source at host 320, which returns 304 a fresh copy of data values to be used for refreshing the cached object (i.e. re-populating the object's output properties) in the cache 300. Whether

a cached copy is already cached or a fresh copy must be retrieved is transparent to the requesting application, which operates as though an actual access (shown in FIG. 3A as imaginary access 325) had been made directly to the back-end data source and a copy had been retrieved in real time.

Since, in lms, a caching policy preferably specifies when the object is to be periodically refreshed, lms is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

The final Office Action asserts in section C, in the fourth full paragraph at page 11, that:

That is, lms teaches if the number of data records updated (col. 15 line 56; updates requested) is not in the fixed range (col. 15 line 58; 5 p.m. and midnight), the update condition setting unit sets the cache update condition (col. 13 line 63; e.g. refresh policy, col. 15 line 64; e.g. update mode) such that the number of date records updated (updated col. 15 line 56; updates requested) fall in the fixed range (col. 15 line 20-33 and line 55-58; e.g. delaying updates to be processed when traffic is low).

This is submitted to be incorrect. The update mode of lms, rather, may vary over *time* for a particular object, not "in a fixed range" as recited in claims 1 and 11. Neither a predetermined period nor a particular time period is a range of data records, contrary to the implication in the final Office Action at page 11. In lms, moreover, the update mode may be set such that delayed updates are selected between the *hours* of 8 a.m. and 5 p.m, not "a number of data records is in a fixed range" as recited in claims 1 and 11. In particular, as described at column 15, lines 52-59:

In a particular system, the update mode may be set such that delayed updates are selected between the hours of 8 a.m. and 5 p.m. because the system is heavily used in that timeframe, as an example, while updates which are requested between midnight and 8 a.m. use the synchronous immediate mode and those requested between 5 p.m. and midnight use the asynchronous immediate mode.

Since, in lms, the update mode may be set such that delayed updates are selected between the hours of 8 a.m. and 5 p.m, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in, for example, claims 1 and 11.

Finally, the final Office Action asserts in section C, and the last full paragraph at page 11, that:

Put another way, lms determines if the updates (e.g. updates from 8 a.m. to 5 p.m.) are not in a fixed range (e.g. 5 p.m. and midnight), then those updates are delayed so that they are updated in ("fall in") that range (e.g. the update condition

is set to delay the updates to when network traffic is low).

This is also submitted to be incorrect. If the updates of lms do not occur between 5 PM and midnight, *delaying* those updates won't make them occur between 5 PM and midnight. Rather, if the interval has passed, there is no way a *delay* will make the interval come back again. Consequently, even under the interpretation of lms adopted by the final Office Action, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in, for example, claims 1 and 11. Further consideration is requested.

**Claim Rejections - 35 U.S.C. § 102:**

Claims 1, 3-5 and 11-14 were rejected under 35 U.S.C. § 102(b) as anticipated by U.S. Patent No. 6,505,200 to lms et al. (hereinafter "lms"). The rejection is traversed to the extent it might apply to the claims as amended. Reconsideration is earnestly solicited.

In the claimed invention, an application server retrieves data from a database management system using a retrieval condition. The retrieval condition is received from a terminal. The application server transmits the data as a retrieval result to the terminal.

The application server has a cache memory that stores the retrieval condition and the retrieval result in a correlated form. The application server also has an update condition setting unit that sets a cache update condition based on a database update condition. The database update condition indicates when the cache memory is to be updated.

The application server also has an update processing unit that determines whether to transmit a retrieval request based on the cache update condition. Finally, the application server has a main controlling unit that sends the retrieval condition to the database management system. The retrieval condition, by the way, is produced based on the retrieval request received from the update processing unit. The main controlling unit also receives the retrieval request of the database from the database management system, and stores the retrieval condition and the retrieval request in the correlated form in the cache memory. The fifth clause of claim 1, and the sixth clause of claim 11, in particular, recite:

A main controlling unit that sends, to the database management system, the retrieval condition that is produced based on the retrieval request received from the update processing unit and receives the retrieval result of a database from the database management system and stores the retrieval condition and the retrieval result in the correlated form in the cache memory.

Ims neither teaches, discloses, or suggests "a main controlling unit that sends, to the database management system, the retrieval condition that is produced based on the retrieval request received from the update processing unit and receives the retrieval result of a database from the database management system and stores the retrieval condition and the retrieval result in the correlated form in the cache memory," as recited in claims 1 and 11.

The application server is characterized in that the update condition setting unit sets the cache update condition by acquiring data updated within a predetermined period from the database, and determines whether a number of data records is in a fixed range. If the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated falls within the fixed range.

The final clauses of claims 1 and 11, in particular, recite substantially:

The update condition setting unit sets the cache update condition by acquiring data updated within a predetermined period from the database, and determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range.

Ims neither teaches, discloses, nor suggests "the update condition setting unit sets the cache update condition by acquiring data updated within a predetermined period from the database, and determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11. Ims, in fact, mentions no "data records" or "fixed range" at all.

The update mode of Ims, rather, may vary over *time* for a particular object, not "in a fixed range" as recited in claims 1 and 11. Neither a predetermined period nor a particular time period is a range of data records, contrary to the implication in the final Office Action at page 4. In particular, as described at column 15, lines 43, 44, and 45:

According to the preferred embodiment, the update mode may vary among the objects of a particular application, and may also vary over time for a particular object.

Since, in Ims, the update mode may vary over time for a particular object, Ims is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update

condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

In lms, moreover, the cache manager uses information about when an update is requested to determine which update *mode* to use, not "whether a number of data records is in a fixed range" as recited in claims 1 and 11. In particular, as described at column 15, lines 45-49:

Preferably, each cached object includes information which can be interrogated by the cache manager when an update is requested. The cache manager then uses the result to determine which update mode to use.

Since, in lms, the cache manager uses information about when an update is requested to determine which update mode to use, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

The cached object of lms, moreover, obtains the current time of day, not "whether a number of data records is in a fixed range" as recited in claims 1 and 11, and based on this result, specifies whether delayed or immediate processing is to be used. In particular, as described at column 15, lines 49-52:

For example, a cached object may include a method that obtains the current time of day, and based on this result, specifies whether delayed or immediate processing is to be used.

Since, in lms, the cached object obtains the current time of day, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

Determining the number of updates requested within a particular *time* period, moreover, does not amount to determining whether the number of data records is in a fixed range, contrary to the assertion at page 4 of the final Office Action. In lms, rather, delayed updates are selected between the hours of 8 a.m. and 5 p.m. because the system is heavily used in that timeframe. In lms, in fact, the update mode may be set such that delayed updates are selected between the *hours* of 8 a.m. and 5 p.m., not "a number of data records is in a fixed range" as recited in claims 1 and 11. In particular, as described at column 15, lines 52-59:

In a particular system, the update mode may be set such that delayed updates are selected between the hours of 8 a.m. and 5 p.m. because the system is

heavily used in that timeframe, as an example, while updates which are requested between midnight and 8 a.m. use the synchronous immediate mode and those requested between 5 p.m. and midnight use the asynchronous immediate mode.

Since, in lms, the update mode may be set such that delayed updates are selected between the hours of 8 a.m. and 5 p.m, lms is not “determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range,” as recited in claims 1 and 11.

lms, moreover, is counting the number of updates requested within a particular time period, and altering the update mode accordingly in order to reduce network traffic. In particular, as described column 15, lines 59-62:

Or, more complex evaluations may be performed such as counting the number of updates requested within a particular time period, and altering the update mode accordingly in order to reduce network traffic.

Since lms is counting the number of updates requested within a particular time period, and altering the update mode accordingly in order to reduce network traffic, lms is not “determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range,” as recited in claims 1 and 11.

In lms, moreover, a cached object is refreshed by executing the retrieval logic in the execution script of the cached object. In particular, as described column 14, lines 51-53:

For example, cached object HAO2 shown at 312 is refreshed by executing the retrieval logic in the execution script of the cached object.

Since, in lms, cached object is refreshed by executing the retrieval logic in the execution script of the cached object, lms is not “determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range,” as recited in claims 1 and 11.

In lms, moreover, the retrieval logic causes a request to be sent to the *back-end* data source, which returns a fresh copy of data values to be used for refreshing the cached object (i.e. re-populating the object's output properties) in the cache. In particular, as described column 14, lines 53-57:

This retrieval logic causes a request to be sent 303 to the back-end data source at host 320, which returns 304 a fresh copy of data values to be used for refreshing the cached object (i.e. re-populating the object's output properties) in the cache 300.

Since, in lms, the retrieval logic causes a request to be sent to the back-end data source, which returns a fresh copy of data values to be used for refreshing the cached object (i.e. re-populating the object's output properties) in the cache, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

In lms, moreover, whether a cached copy is already cached or a fresh copy must be retrieved is *transparent* to the requesting application, which operates as though an actual access had been made directly to the back-end data source and a copy had been retrieved in real time. In particular, as described column 14, lines 57-62:

Whether a cached copy is already cached or a fresh copy must be retrieved is transparent to the requesting application, which operates as though an actual access (shown in FIG. 3A as imaginary access 325) had been made directly to the back-end data source and a copy had been retrieved in real time.

Since, in lms, whether a cached copy is already cached or a fresh copy must be retrieved is transparent to the requesting application, which operates as though an actual access had been made directly to the back-end data source and a copy had been retrieved in real time, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

Finally, in lms, the refresh policy of each cached object may be periodically *evaluated* to determine whether the associated cached objects have become invalid. In particular, as described column 14, lines 62-67:

In an optional feature of this aspect, the refresh policy of each cached object may be periodically evaluated (e.g. at periodic time intervals, or in response to predetermined events) to determine whether the associated cached objects have become invalid.

Since, in lms, the refresh policy of each cached object may be periodically evaluated to determine whether the associated cached objects have become invalid, lms is not "determining whether a number of data records is in a fixed range, and if the number of data records updated

is not in the fixed range, the update condition setting unit sets the cache update condition such that the number of data records updated fall in the fixed range," as recited in claims 1 and 11.

The second clause of claim 1, and the third clause of claim 11, recites:

A cache memory that stores in a correlated form the retrieval condition and the retrieval result.

Ims neither teaches, discloses, nor suggests "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11. Ims, rather, is automatically *synchronizing* one data store with another data store, even though the two stores may not share a common format. In particular, as described column 4, lines 52-56:

An object of the present invention is to provide a technique whereby one data store can be automatically synchronized with another data store, even though the two stores may not share a common format.

Since Ims is automatically synchronizing one data store with another data store, even though the two stores may not share a common format, Ims is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

In Ims, moreover, the cached objects *component* provides for storing, and for automatically refreshing, cached objects. In particular, as described at column 9, lines 53-59:

The present invention uses a "cached objects" component (referred to equivalently herein as a "cache manager") which caches objects that are used by middleware applications to interact with back-end data sources, where the middleware application functions as a surrogate for a requesting client application. The cached objects component provides for storing, and for automatically refreshing, these objects.

Since, in Ims, the cached objects component provides for storing, and for automatically refreshing, cached objects, Ims is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

In Ims, moreover, a cached object stored by the cached objects component has a set of *input* properties and a set of *output* properties, not a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11. In particular, as described at column 10, lines 21-25:

Each object stored by the cached objects component has a set of input properties and a set of output properties (which might not be set when the object is cached) representing the information to and from the object's corresponding back-end data source.

Since, in lms, a cached object stored by the cached objects component has a set of input properties and a set of output properties, lms is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

In lms, moreover, a cached object includes processing logic which describes how the object interacts with the *back-end* data source, and an execution method which invokes this processing logic, not a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11. In particular, as described at column 10, lines 21-28:

Further, each cached object preferably includes processing logic which describes how the object interacts with the back-end data source, and an execution method which invokes this processing logic.

Since, in lms, a cached object may include processing logic which describes how the object interacts with the back-end data source, and an execution method which invokes this processing logic, lms is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

Finally, in lms, the input properties and values thereof are used as an index to *locate* objects stored in the cached objects component, not as a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11. It is the stored objects *themselves*, rather, that may be categorized as either read-access (RA) or write-access (WA). In particular, as described at column 10, lines 28-32:

The input properties and values thereof are used as an index to locate objects stored in the cached objects component, where those stored objects may be categorized as either read-access (RA) or write-access (WA).

Since, in lms, the input properties and values thereof are used as an index to locate objects stored in the cached objects component, lms is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

Furthermore, in lms, a caching policy specifies when the object is to be *periodically* refreshed, such as at a particular time of day; upon occurrence of a specified event; upon an elapsed time since the last refresh, rather than according to a "retrieval condition and the retrieval result," as recited in claims 1 and 11. In particular, as described at column 13, lines 56-62:

The caching policy of an RA object preferably specifies when the object is to be periodically refreshed. For example, an object may be refreshed at a particular time of day; upon occurrence of a specified event; upon an elapsed time since the last refresh; etc. This refresh process thereby ensures that a relatively up-to-

date version of the back-end data is being used when responding to client read requests.

Since, in *lms*, a caching policy preferably specifies when the object is to be periodically refreshed, *lms* is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

In *lms*, moreover, there is retrieval *logic* in the execution script of the cached object, not a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11. In particular, as described at column 14, lines 51-62:

For example, cached object HAO2 shown at 312 is refreshed by executing the retrieval logic in the execution script of the cached object. This retrieval logic causes a request to be sent 303 to the back-end data source at host 320, which returns 304 a fresh copy of data values to be used for refreshing the cached object (i.e. re-populating the object's output properties) in the cache 300. Whether a cached copy is already cached or a fresh copy must be retrieved is transparent to the requesting application, which operates as though an actual access (shown in FIG. 3A as imaginary access 325) had been made directly to the back-end data source and a copy had been retrieved in real time.

Since, in *lms*, a caching policy preferably specifies when the object is to be periodically refreshed, *lms* is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

Finally, in *lms*, refresh *logic* is stored with or associated with selected replicated read-access objects, and update *logic* is stored with or associated with selected replication write-access objects, not a "retrieval condition" or a "retrieval result," as recited in claims 1 and 11. In particular, as described at column 5, lines 64-67, continuing at column 6, lines 1 and 2:

Performing the replication may further comprise executing the refresh logic stored with or associated with selected replicated read-access objects for which the queued refresh requests are queued, and executing the update logic stored with or associated with selected replication write-access objects for which the queued update requests are queued.

Since, in *lms*, refresh logic is stored with or associated with selected replicated read-access objects, and update logic is stored with or associated with selected replication write-access objects, *lms* is not setting "a cache memory that stores in a correlated form the retrieval condition and the retrieval result," as recited in claims 1 and 11.

The fourth clause of claim 1, and the fifth clause of claim 11, recites:

An update processing unit that determines whether to transmit a retrieval request based on the cache update condition.

lms neither teaches, discloses, nor suggests "an update processing unit that determines whether to transmit a retrieval request based on the cache update condition," as recited in claims 1 and 11. In lms, rather, there is retrieval *logic* in the execution script of the cached object, not a "retrieval condition," as discussed above.

Moreover, in lms, the object's update operation is being applied to the *back-end data source* at host 340, not the cached object. In particular, as described at column 16, lines 58-66:

When the update policy of an object having elements queued in its update queue 352 is triggered (for example, reaching a certain time of day when low-priority batched mode requests are to be processed), the updates from the queue 352 are processed by executing the update script of the corresponding object. This execution causes the object's update operation to be applied 353 to the back-end data source at host 340, by execution of the object's script using the input property values from the queued element.

Since, in lms, the object's update operation is being applied to the back-end data source at host 340, not the cached object, lms has no "update processing unit that determines whether to transmit a retrieval request based on the cache update condition," as recited in claims 1 and 11. Claims 1 and 11 are submitted to be allowable. Withdrawal of the rejection of claims 1 and 11 is earnestly solicited.

Claims 3, 4, and 5 depend from claim 1 and add further distinguishing elements, while claims 12 and 13 depend from claim 11 and add further distinguishing elements. Claims 3, 4, 5, 12, and 13 are thus also submitted to be allowable. Withdrawal of the rejection of claims 3, 4, 5, 12, and 13 is also earnestly solicited.

Claim 14:

The last clause of claim 14 recites:

Setting the cache update condition by acquiring data updated within a predetermined period from the database, and determining whether a number of data records updated within the predetermined period is in a fixed range, and if the number of data records updated is not in the fixed range, the setting includes setting the cache update condition such that the number of data records updated fall in the fixed range.

lms neither teaches, discloses, nor suggests " setting the cache update condition by acquiring data updated within a predetermined period from the database, and determining whether a number of data records updated within the predetermined period is in a fixed range, and if the number of data records updated is not in the fixed range, the setting includes setting the cache

update condition such that the number of data records updated fall in the fixed range," as discussed above with respect to the rejection of claim 1.

The second clause of claim 14 recites:

Storing a retrieval condition received from a terminal and a retrieval result retrieved using the retrieval request in a correlated form in a cache memory.

Ims neither teaches, discloses, nor suggests "a retrieval condition received from a terminal and a retrieval result retrieved using the retrieval request in a correlated form in a cache memory," as discussed above with respect to the rejection of claim 1.

The fifth clause of claim 14 recites:

Determining whether to transmit a retrieval request based on the cache update condition.

Ims neither teaches, discloses, nor suggests "determining whether to transmit a retrieval request based on the cache update condition," as discussed above with respect to the rejection of claim 1. Claim 14 is thus also submitted to be allowable for at least those reasons discussed above respect to the rejections of claims 1 and 11. Withdrawal of the rejection of claim 14 is earnestly solicited.

New Claim 15:

The second clause of claim 15 recites:

A cache memory that stores therein in a correlated form a retrieval expression and a retrieval result.

Ims neither teaches, discloses, or suggests "a cache memory that stores therein in a correlated form a retrieval expression and a retrieval result," as discussed above with respect to the rejection of claims 1 and 11. The second clause of claim 15 recites:

A main controlling unit that sends, to a database management system, the retrieval expression that is produced based on a retrieval condition received from a client terminal or read from the cache memory at a given interval, receives the retrieval result of a database by the retrieval expression from the database management system, and stores the retrieval expression and the retrieval result in the correlated form in the cache memory.

Ims neither teaches, discloses, or suggests "a main controlling unit that sends, to a database management system, the retrieval expression that is produced based on a retrieval condition received from a client terminal or read from the cache memory at a given interval, receives the retrieval result of a database by the retrieval expression from the database

management system, and stores the retrieval expression and the retrieval result in the correlated form in the cache memory," as discussed above with respect to the rejection of claims 1 and 11. Claim 15 is thus believed to be allowable.

New Claim 16:

The second clause of claim 16 recites:

Storing a retrieval expression and a retrieval result in a correlated form in a cache memory.

Ims neither teaches, discloses, or suggests "storing a retrieval expression and a retrieval result in a correlated form in a cache memory," as discussed above with respect to the rejection of claims 1 and 11.

The third clause of claim 16 recites:

Sending, to a database management system, the retrieval expression that is produced based on a retrieval condition received from a client terminal or read from the cache memory at a given interval.

Ims neither teaches, discloses, or suggests "sending, to a database management system, the retrieval expression that is produced based on a retrieval condition received from a client terminal or read from the cache memory at a given interval," as discussed above with respect to the rejection of claims 1 and 11.

The fifth clause of claim 16 recites:

Storing the retrieval expression and the retrieval result in the correlated form in the cache memory.

Ims neither teaches, discloses, or suggests "storing the retrieval expression and the retrieval result in the correlated form in the cache memory," as discussed above with respect to the rejection of claims 1 and 11. Claim 16 is thus believed to be allowable.

**Conclusion:**

Accordingly, in view of the reasons given above, it is submitted that all of claims 1, 3-5 and 11-16 are allowable over the cited references. Allowance of all claims 1, 3-5 and 11-16 and of this entire application is therefore respectfully requested.

If there are any formal matters remaining after this response, the Examiner is invited to telephone the undersigned to attend to these matters.

Application Serial No. 10/766,839  
Submission with RCE filed August 27, 2009 (Monday)  
Reply to final Office Action mailed November 26, 2008

If there are any additional fees associated with filing of this Amendment, please charge them to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: April 27, 2009

By: /Thomas E. McKiernan/  
Thomas E. McKiernan  
Registration No. 37,889

1201 New York Ave, N.W., 7th Floor  
Washington, D.C. 20005  
Telephone: (202) 434-1500  
Facsimile: (202) 434-1501